

AMENDMENTS TO THE CLAIMS

1. (Previously Presented) An in-order multi-threading processor, comprising:
 - a first instruction fetch unit to receive a first thread and a second instruction fetch unit to receive a second thread;
 - an execution unit to execute said first thread and said second thread in parallel;

and

 - a multi-thread scheduler coupled to said first instruction fetch unit, said second instruction fetch unit, and said execution unit, wherein said multi-thread scheduler is to determine the width of said execution unit.
2. (Previously Presented) An in-order multi-threading processor as recited in claim 1, wherein said multi-thread scheduler unit determines whether said execution unit is to execute said first thread and said second thread in parallel depending on the width of said execution unit.
- 3-4 (Cancelled).
5. (Previously Presented) An in-order multi-threading processor as recited in claim 2, wherein said execution unit executes a third thread and a fourth thread in series.

6. (Previously Presented) An in-order multi-threading processor as recited in claim 2, wherein said first thread and said second thread are compiled to have instruction level parallelism.

7. (Previously Presented) An in-order multi-threading processor as recited in claim 6, further comprising:

a first instruction decode unit coupled between said first instruction fetch unit and said multi-thread scheduler; and

a second instruction decode unit coupled between said second instruction fetch unit and said multi-thread scheduler.

8. (Previously Presented) An in-order multi-threading processor as recited in claim 4, wherein said execution unit executes only two threads in parallel.

9. (Previously Presented) A computer implemented method, comprising:

determining whether an in-order multi-threading processor is wide enough to execute a first thread and a second thread in parallel; and

executing said first thread and said second thread in parallel if said in-order multi-threading processor is wide enough to execute said first thread and said second thread in parallel.

10. (Previously Presented) The method as recited in claim 9, further comprising executing said first thread and said second thread in series if said in-order multi-threading processor is not wide enough.

11. (Cancelled).

12. (Previously Presented) The method as recited in claim 9, further comprising compiling the first thread and the second thread, wherein the first thread and the second thread have instruction level parallelism.

13. (Previously Presented) The method as recited in claim 12, wherein said multi-threading processor executes only two threads in parallel.

14. (Previously Presented) The method as recited in claim 13, further comprising:
fetching said first thread and said second thread; and
decoding said first thread and said second thread.

15. (Previously Presented) A set of instructions residing in a storage medium, said set of instructions to be executed by an in-order multi-threading processor for searching data stored in a mass storage device comprising:
determining whether said in-order multi-threading processor is wide enough to execute a first thread and a second thread in parallel; and

executing said first thread and said second thread in parallel if said multi-threading processor is wide enough to execute said first thread and said second thread in parallel.

16. (Previously Presented) A set of instructions as recited in claim 15, further comprising executing said first thread and said second thread in series if said in-order multi-threading processor is not wide enough.

17. (Cancelled).

18. (Previously Presented) A set of instructions as recited in claim 16, further comprising compiling said first thread and said second thread, wherein said first thread and said second thread have instruction level parallelism.

19. (Previously Presented) A set of instructions as recited in claim 18, wherein said in-order multi-threading processor executes only two threads in parallel.

20. (Previously Presented) A set of instructions as recited in claim 19, further comprising:

fetching said first thread and said second thread; and
decoding said first thread and said second thread.

21. (Previously Presented) A system comprising:
 - a memory to store a set of instructions; and
 - an in-order processor coupled to the memory to execute said set of instructions, said in-order processor with a first instruction fetch unit to receive a first thread, a second instruction fetch unit to receive a second thread, an execution unit to execute said first thread and said second thread, and a multi-thread scheduler coupled to said first instruction fetch unit, said second instruction fetch unit, and said execution unit, wherein said multi-thread scheduler is to determine the width of said execution unit.
22. (Previously Presented) The system of claim 21, wherein said multi-thread scheduler unit determines whether said execution unit is to execute said first thread and said second thread in parallel depending on the width of said execution unit.
23. (Cancelled).
24. (Previously Presented) The system of claim 22, wherein said execution unit executes said first thread and said second thread in parallel.
25. (Previously Presented) The system of claim 22, wherein said execution unit executes said first thread and said second thread in series.

26. (Previously Presented) The system of claim 22, wherein said first thread and said second thread are compiled to have instruction level parallelism.

27. (Previously Presented) The system of claim 26, further comprising:
a first instruction decode unit coupled between said first instruction fetch unit and
said multi-thread scheduler; and
a second instruction decode unit coupled between said second instruction fetch
unit and said multi-thread scheduler.

28. (Previously Presented) The system of claim 24, wherein said execution unit
executes only two threads in parallel.